# Final Project

---

## Project Description:

---

A leading music-catering company is planning to analyse large amount of data received from varieties of sources, namely mobile app and website to track the behaviour of users, classify users, calculate royalties associated with the song and make appropriate business strategies. The file server receives data files periodically at every 3 hours.

**As part of the project need to do the below things:**

1. Data Ingestion and Initial Validation - taken care by generate_web_data.py and generate_mob_data.py
2. Data Enrichment - taken care by data_enrichment.sh
3. Post Enrichment - taken care by data_formattig.sh
4. Data Analysis - taken care by data_analysis.sh
5. Post Data Analysis - taken care by datapostanalysis.sh

**Data Ingestion and Initial Validation: -**

---

**Rules for data ingestion and data filtering:**

1.Data coming from mobile applications reside in /data/mob and has csv format.

2. Data coming from web applications reside in /data/web and has xml format.

3. Data files come every 3 hours.

4. All the timestamp fields in data coming from web application is of the format YYYY-MM-DD HH:MM:SS.

5. All the timestamp fields in data coming from mobile application is a long integer interpreted as UNIX timestamps.

6. Finally, all timestamps must have the format of a long integer to be interpreted as UNIX timestamps.

7. If both like and dislike are 1, consider that record to be invalid.
8. If any of the fields from User_id, Song_id, Timestamp, Start_ts, End_ts, Geo_cd is NULL or absent, consider that record to be invalid.
9.If Song_end_type is NULL or absent, treat it to be 3 Create a temporary identifier for all the data files received in the last 3 hours (may be an integer batch_id which is auto incremented or a string obtained aer combining current date and current hour, to keep track of valid.

**Data Enrichment:**

**----------**

**Rules for data enrichment**

1. If any of like or dislike is NULL or absent, consider it as 0.
2. If fields like Geo_cd and Artist_id are NULL or absent, consult the lookup tables for fields Station_id and Song_id respectively to get the values of Geo_cd and Artist_id.
If corresponding lookup entry is not found, consider that record to be invalid.

**Post Enrichment:**

**----------**

Move all valid records in /hadoop/processing_dir in HDFS and invalid records in Local File System at /usr/invalid directory.
Maintain a copy of valid records in /usr/validated in Local File System. Run a cleaner everyday to clean validated files which are more than 7 days old.

**Data Analysis:**

**----------**

It is not only the data which is important, rather it is the insight it can be used to generate important. Once we have made the data ready for analysis, we have to perform below analysis on a daily basis.

Determine total duration of songs played by each type of user, where type of user can be 'subscribed' or ' unsubscribed'. An unsubscribed user is the

one whose record is either not present in Subscribed_users lookup table or has subscription_end_date earlier than the timestamp of the song played by him. Determine top 10 connected artists. Connected artists are those whose songs are most listened by the unique users who follow them. Determine top 10 songs who have generated the maximum revenue. Royalty applies to a song only if it was liked or was completed successfully or both. 5. Determine top 10 unsubscribed users who listened to the songs for the longest duration. Station_id Station_Geo_Map Artist_id Song_id Song_Artist_Map

**Post Analysis:**

**----------**

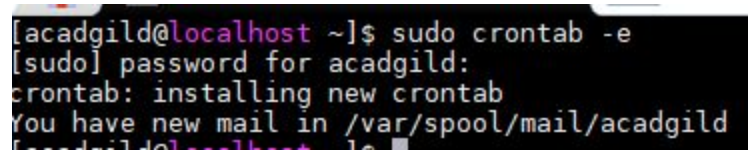
Once the analysis is complete, multiple actions can be taken place later on. It includes: 1. Moving result of analysis to the RDMS for data storage and quick retrieval. 2. Form visualisations on the top of analysed data. 3. Send data to data science or machine learning pipelines for further forecast.

**Project explanation:**

**----------**

**PREREQUISITES:**

1. Start the hive meta store as part of the project before running the master_project_master.sh by giving the command **hive --service  metastore**
2. Edit the crontab to start the project execution for every 3 hours as shown below:
3. Give the password as acadgild and open the file and add the script that run the shell script every 3 hours





Each part of the project execution mentioned above as part of the projects are executed with each separate shell script files. There is one master shell

script namely **music_project_master.sh** which calls all other script files and required commands to perform each stage of project execution.

**The list of files used are :**

**----------**

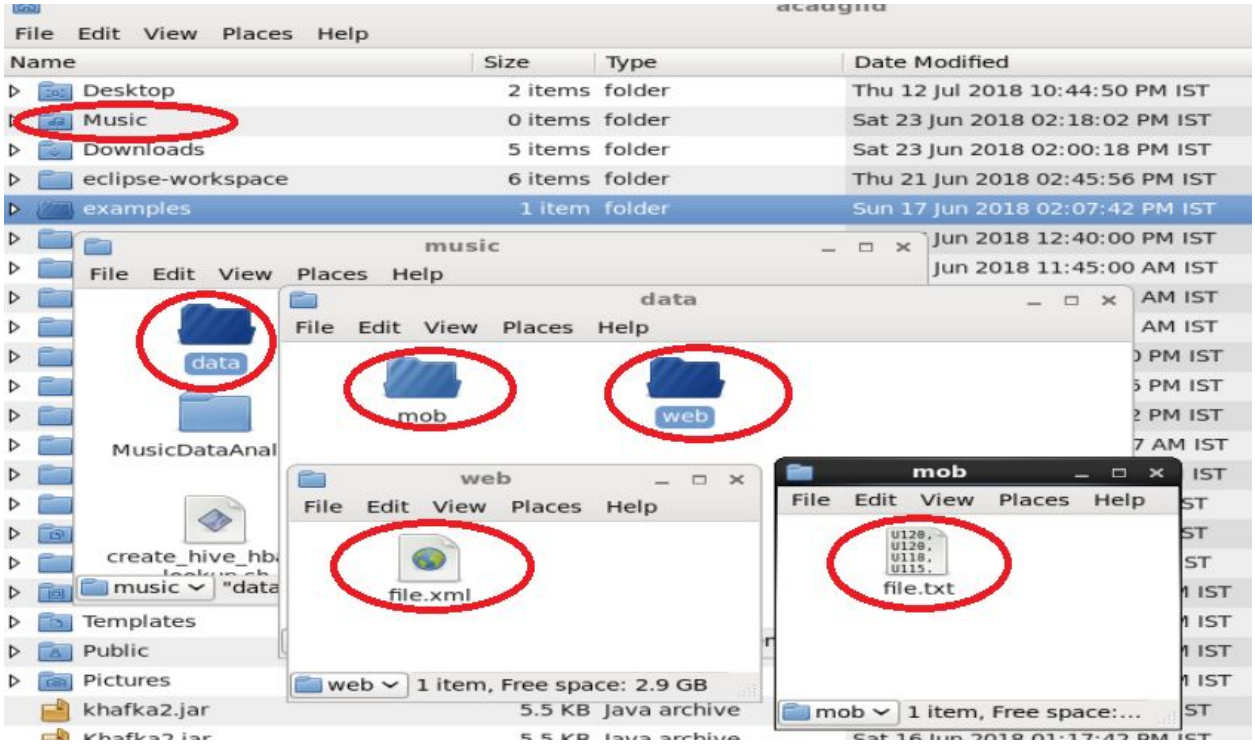
1. **Music_project_master.sh:** - This is the main file which calls all the scripts to include different parts of the project executions like data formatting, data enrichment and data analysis. Let's see them in detail below.

**Music_project_master.sh scripts does the below things:**

**----------**

1. Removes the path given below using the below command
   **rm -r /home/acadgild/examples/music/data/web**
   **rm -r /home/acadgild/examples/music/data/mob**
2. Creates the below folders once again using below commands:
   **mkdir -p /home/acadgild/examples/music/data/web**
   **mkdir -p /home/acadgild/examples/music/data/mob**

   **Snapshot for above folder that is created:**
   **----------**




3. Calls the below python script to create the web and mobile data in respective folders. The below script creates input files using random number

**python /home/acadgild/examples/music/generate_web_data.py**
**python /home/acadgild/examples/music/generate_mob_data.py**

The above 2 scripts create the input files namely file.xml and file.txt one for web and the other for mobile data respectively. The above steps include data ingestion

4. Calls **start-daemons.sh** to start all the hadoop daemons and checks the started daemons as well using **JPS** command.

5. Calls the **populate-lookup.sh** shell script to upload the look up tables in hbase.

6. Calls data enrichment shell script(**data_enrichment_filtering_schema.sh**) for creating hive tables on top of hbase tables for data enrichment and filtering.

7. Calls data formatting shell script(**dataformatting.sh**) to perform some data formatting which does hive table with hbase mapping.
8. Calls data enrichment shell script( **data_enrichment.sh**) to do data enrichment that is needed as per requirement."
9. Calls data analysis shell script(**data_analysis.sh**) to do data analysis which will get list of details as required by the project.
10.Calls postdatanalysis.sh to perform the post data analysis as per project requirement.
Lets see each script files in detail along with output screenshots.
**All the script files will be uploaded separately in github.**

**2. Generate_web_data.py:**

**----------**

This script creates file.xml file using unix commands using rand keyword. This file will be the web data input.

**Snapshot of file.xml that was created as part of script execution:**

**----------**

```
<records>
<record>
<user_id>U117</user_id>
<song_id>S208</song_id>
<artist_id>A300</artist_id>
<timestamp>2016-06-09 22:12:36</timestamp>
<start_ts>2017-05-09 08:09:22</start_ts>
<end_ts>2016-07-10 01:38:09</end_ts>
<geo_cd>AU</geo_cd>
<station_id>ST405</station_id>
<song_end_type>1</song_end_type>
<like>1</like>
<dislike>0</dislike>
</record>
<record>
<user_id>U114</user_id>
<song_id>S201</song_id>
<artist_id>A305</artist_id>
<timestamp>2016-06-09 22:12:36</timestamp>
<start_ts>2017-05-09 08:09:22</start_ts>
<end_ts>2016-05-10 12:24:22</end_ts>
<geo_cd>A</geo_cd>
<station_id>ST404</station_id>
<song_end_type>2</song_end_type>
<like>1</like>
<dislike>1</dislike>
</record>
```

**3. Generate_mob_data.py:**

**----------**

This scripts create file.txt file using unix commands using rand keyword. This file will be the mobile data input.

**Snapshot of file.xml that was created as part of script execution:**

**----------**

```
U120,S205,A303,1475130523,1475130523,1485130523,AP,ST401,2,0,0
U120,S210,A300,1465130523,1465230523,1475130523,E,ST413,2,0,1
U118,S202,A304,1465230523,1465230523,1465130523,U,ST411,2,1,1
U115,S203,A304,1465230523,1465130523,1465230523,A,ST404,2,0,1
U113,S203,A303,1465230523,1485130523,1465130523,A,ST415,3,0,1
,S205,A305,1475130523,1475130523,1485130523,E,ST406,3,1,1
U105,S207,A300,1465230523,1475130523,1465130523,E,ST414,0,0,1
U101,S209,A305,1465230523,1485130523,1465230523,AP,ST414,3,0,1
U105,S202,A305,1475130523,1465230523,1485130523,,ST405,3,1,0
U105,S203,,1475130523,1475130523,1465230523,AU,ST415,2,1,0
U114,S205,A304,1465130523,1485130523,1465130523,AP,ST402,2,1,1
U118,S200,A303,1465230523,1465130523,1465130523,AU,ST410,0,1,0
U116,S202,A305,1495130523,1465230523,1485130523,A,ST409,0,1,0
U110,S206,A304,1465230523,1475130523,1475130523,A,ST406,3,0,1
U104,S206,A301,1465230523,1475130523,1485130523,U,ST404,2,1,0
U112,S206,A300,1495130523,1465130523,1465230523,A,ST413,3,1,1
U100,S200,A300,1465130523,1475130523,1485130523,U,ST401,3,1,0
U111,S210,A303,1475130523,1475130523,1465230523,U,ST406,0,1,0
U107,S202,A305,1475130523,1485130523,1465230523,A,ST402,0,1,1
U112,S201,A301,1475130523,1465130523,1485130523,A,ST404,1,0,0
```

**4. Start-daemons.sh:**

**----------**

This script starts the hadoop daemons using the commands

**Start-dfs.sh**

**Start-yarn.sh**

**Start-hbase.sh**

**5. Populate-lookup.sh:**

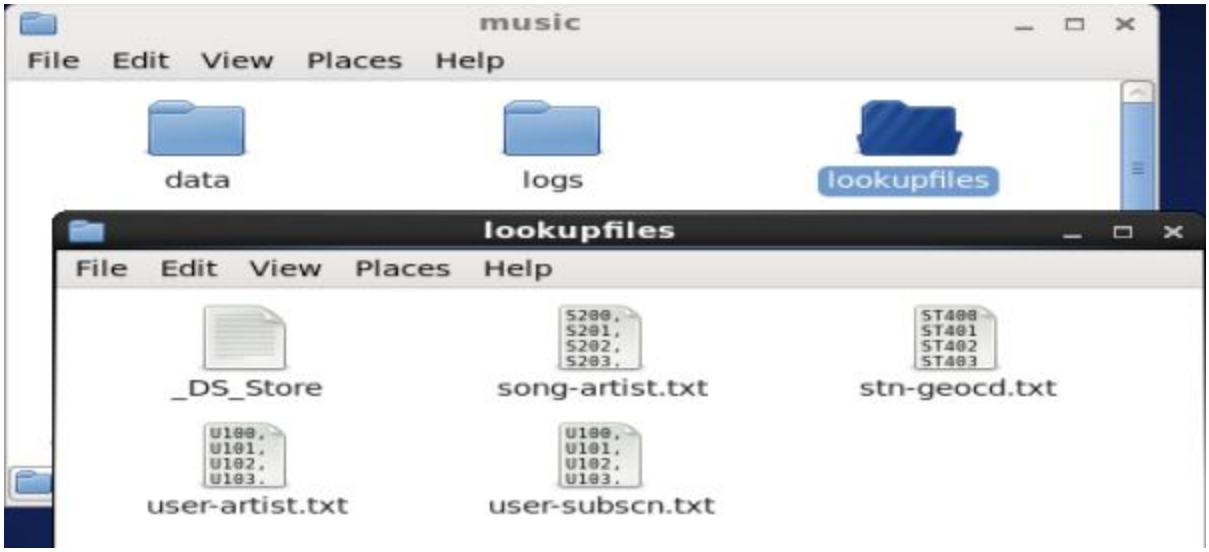
**----------**

This scripts checks if the below mentioned hbase tables are already existing in the hbase, if yes it disables and deletes the tables and recreate new one.

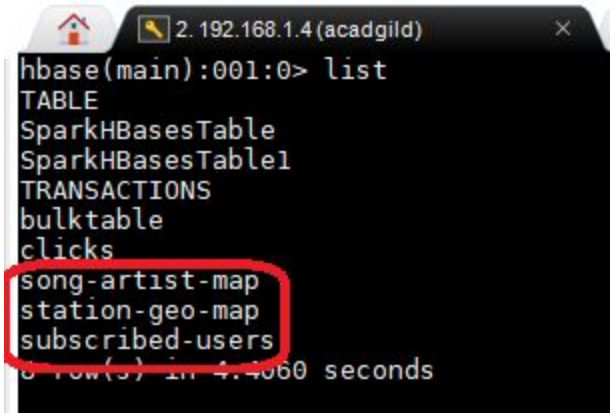
There are 3 text files that are available inside lookupfiles folder which is used for creating hbase tables. The screen shot of the folder containing the files is shown below.

**Screenshot of the Lookup folder:**

---




**Below are the 3 hbase tables that will be created as part of the populate-lookup script:**



**screenshot of the hbase tables that are created as part of the script:**

---

## Song-artist-map and station-geo-map hbase tables:

```
base(main):003:0> scan 'song-artist-map'
ROW                                   COLUMN+CELL
S200                                  column=artist:artistid, timestamp=1529740856201, value=A300
S201                                  column=artist:artistid, timestamp=1529740864427, value=A301
S202                                  column=artist:artistid, timestamp=1529740875214, value=A302
S203                                  column=artist:artistid, timestamp=1529740886067, value=A303
S204                                  column=artist:artistid, timestamp=1529740896632, value=A304
S205                                  column=artist:artistid, timestamp=1529740907225, value=A301
S206                                  column=artist:artistid, timestamp=1529740917670, value=A302
S207                                  column=artist:artistid, timestamp=1529740928253, value=A303
S208                                  column=artist:artistid, timestamp=1529740938909, value=A304
S209                                  column=artist:artistid, timestamp=1529740949503, value=A305
```

```
hbase(main):004:0> scan 'station-geo-map'
ROW                                   COLUMN+CELL
 ST400                                column=geo:geo_cd, timestamp=1529740772972, value=A
 ST401                                column=geo:geo_cd, timestamp=1529740779398, value=AU
 ST402                                column=geo:geo_cd, timestamp=1529740784946, value=AP
 ST403                                column=geo:geo_cd, timestamp=1529740791294, value=J
 ST404                                column=geo:geo_cd, timestamp=1529740795947, value=E
 ST405                                column=geo:geo_cd, timestamp=1529740801962, value=A
 ST406                                column=geo:geo_cd, timestamp=1529740808769, value=AU
 ST407                                column=geo:geo_cd, timestamp=1529740813679, value=AP
 ST408                                column=geo:geo_cd, timestamp=1529740819095, value=E
 ST409                                column=geo:geo_cd, timestamp=1529740824709, value=E
 ST410                                column=geo:geo_cd, timestamp=1529740829989, value=A
 ST411                                column=geo:geo_cd, timestamp=1529740835190, value=A
 ST412                                column=geo:geo_cd, timestamp=1529740841148, value=AP
 ST413                                column=geo:geo_cd, timestamp=1529740846783, value=J
 ST414                                column=geo:geo_cd, timestamp=1529740851342, value=E
15 row(s) in 0.0620 seconds
```

## Subscribed-users hbase table:

```
hbase(main):005:0> scan 'subscribed-users'
ROW                                   COLUMN+CELL
 U100                                 column=subscn:enddt, timestamp=1529740971481, value=1465130523
 U100                                 column=subscn:startdt, timestamp=1529740960576, value=1465230523
 U101                                 column=subscn:enddt, timestamp=1529740992782, value=1475130523
 U101                                 column=subscn:startdt, timestamp=1529740982056, value=1465230523
 U102                                 column=subscn:enddt, timestamp=1529741014522, value=1475130523
 U102                                 column=subscn:startdt, timestamp=1529741003552, value=1465230523
 U103                                 column=subscn:enddt, timestamp=1529741036558, value=1475130523
 U103                                 column=subscn:startdt, timestamp=1529741025413, value=1465230523
 U104                                 column=subscn:enddt, timestamp=1529741058823, value=1475130523
 U104                                 column=subscn:startdt, timestamp=1529741047232, value=1465230523
 U105                                 column=subscn:enddt, timestamp=1529741080803, value=1475130523
 U105                                 column=subscn:startdt, timestamp=1529741069789, value=1465230523
 U106                                 column=subscn:enddt, timestamp=1529741102789, value=1485130523
 U106                                 column=subscn:startdt, timestamp=1529741091797, value=1465230523
 U107                                 column=subscn:enddt, timestamp=1529741124002, value=1455130523
```

**Song-artist-map -** Contains mapping of song_id with artist_id alongwith royalty associated with each play of the song

**Station-geo-map -** Contains mapping of a geo_cd with station_id.

**Subscribed-users -** Contains user_id, subscription_start_date and subscription_end_date. Contains details only for subscribed users.

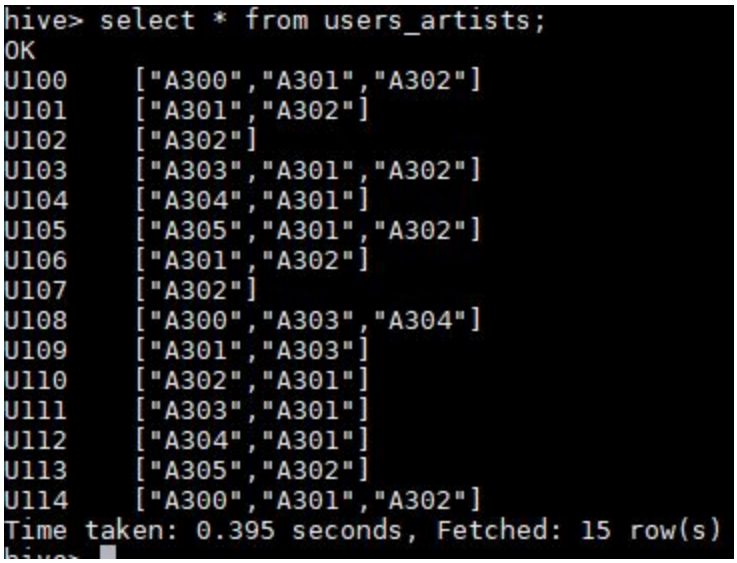
After creating the above 3 hbase tables , it creates a hive table user-artist using user-artist.txt file through user-artist.hql file.
User-artist Contains an array of artist_id(s) followed by a user_id

**Snapshot showing the created hive table user_artists:**

---


**Snapshot showing the data in the users_artists table:**

---



**6. Data_enrichment_filtering_schema.sh:**

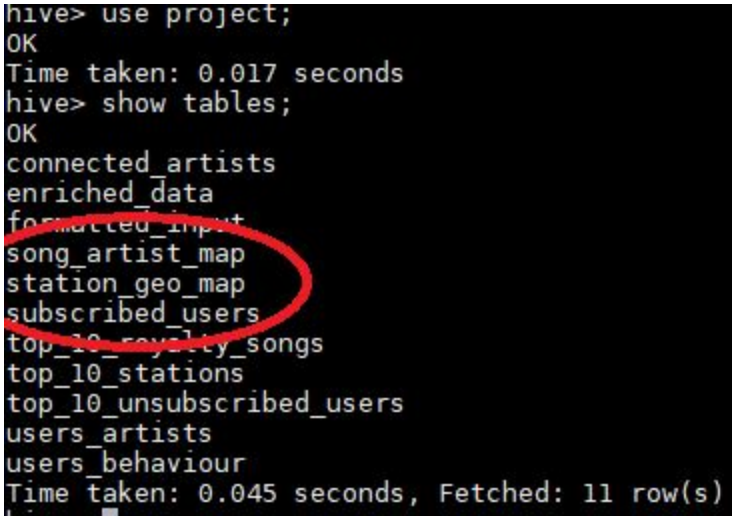
**----------**

This script calls create_hive_hbase_lookup.hql and creates hive tables using hbase tables that was created in previous step.

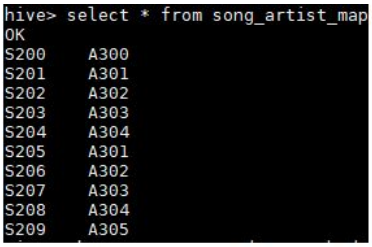
**Tables created in hive are:**

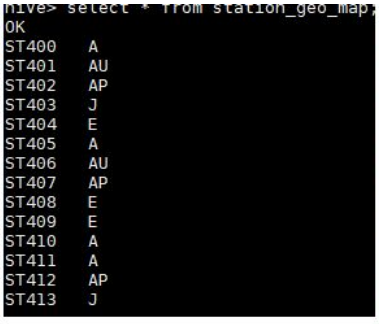
Song_artist_map
Station_geo_map
Subscribed_users

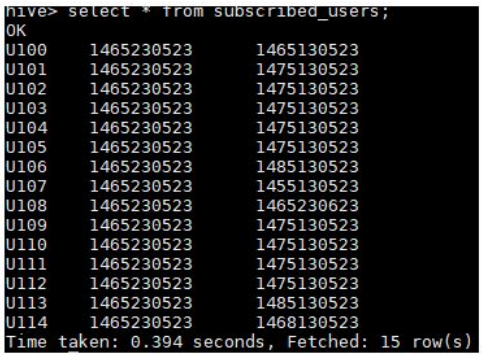
**Snapshot of the hive tables that are created:**




**Snapshot of the data in the tables(song_artist_map, station_geo_map,subscribed_users):**

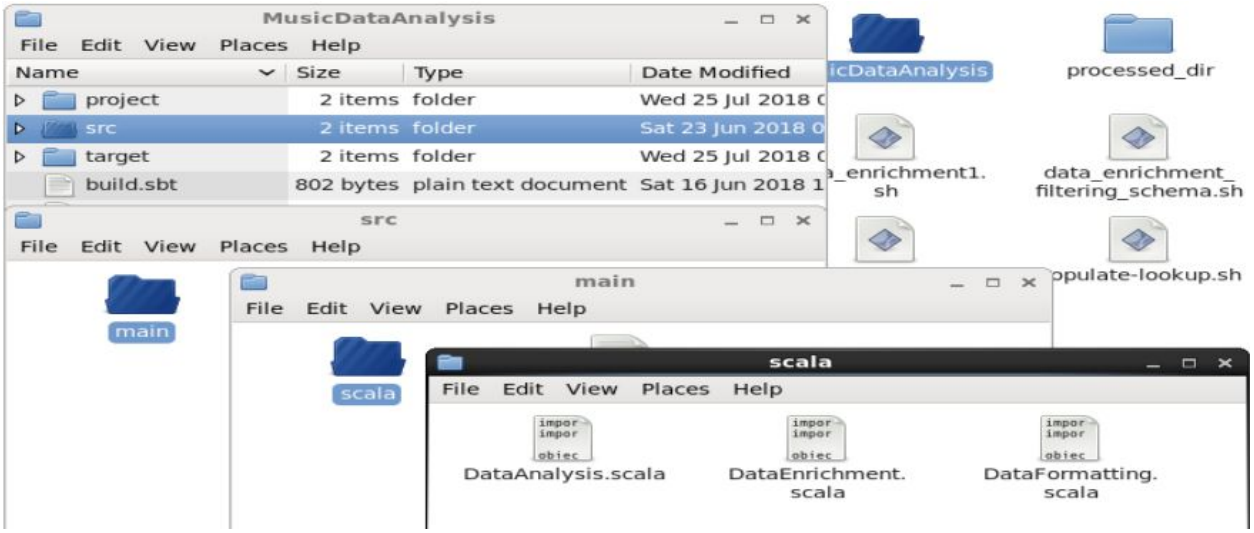


**Common steps for (DataFormatting, DataEnrichment and DataAnalysis):**

For the next 3 tasks(data formatting, data enrichment and data analysis) there are 3 scala files which does the respective tasks. A new folder (**MusicDataAnalysis/src/main/scala**) is created as shown below and 3 scala files are moved to this folder.




In order to call the scala files to do the respective tasks we need to create a package by giving the below command

**Sbt -v package**

The above command creates a jar file with all the 3 scala files.

This jar file is submitted to spark as part of the each shell script.

**7. Dataformatting.sh:**

**----------**

This will remove the folder MusicDataAnalysis/project and MusicDataAnalysis/target folders and recreates them while executing the sbt -v package . After which a jar file is created by compiling all the 3 scala files and is submitted to spark using spark submit command.

The DataFormatting.scala file will create a new hive table(formatted_Input) using mobile data(file.xml) and web data(file.txt) using batchId as partition column.

**Snapshot on the newly created table formatted_input:**

---

```
hive> show tables;
OK
connected_artists
enriched_data
formatted_input
song_artist_map
station_geo_map
subscribed_users
top_10_royalty_songs
top_10_stations
top_10_unsubscribed_users
users_artists
users_behaviour
Time taken: 0.045 seconds, Fetched: 11 row(s)
hive>
```

**Sample output from the fomatted_input file:**

---

```
hive> select * from formatted_input;
OK
U120    S205    A303    1475130523      1475130523      1485130523      AP      ST401   2       0       0       1
U120    S210    A300    1465130523      1465230523      1475130523      E       ST413   2       0       1       1
U118    S202    A304    1465230523      1465230523      1465130523      U       ST411   2       1       1       1
U115    S203    A304    1465230523      1465130523      1465230523      A       ST404   2       0       1       1
U113    S203    A303    1465230523      1485130523      1465130523      A       ST415   3       0       1       1
```

**DataEnrichment.sh:**

---

This script will create processed_dir folder under which it creates 2 more folder valid and invalid.It puts the valid record that is data having pass status in the table into valid folder and the data having fail status in the table into invalid folder. Also this script cleans the records that are 7 days old.

DataEnrichment.scala file will create a new hive table named(Enriched data) from the formatted_input hive table that was created in previous step.

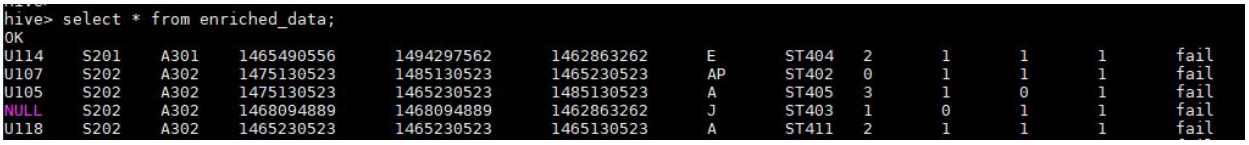
It also does joining from the look up tables station_geo_map, song_artist_map to fill up the empty columns in the new table that is created. example if any of the columns like song_id user_id and few other columns are NULL then it updates the status as fail

**Snapshot of the table created:**

---


**Sample output from the enriched data table:**

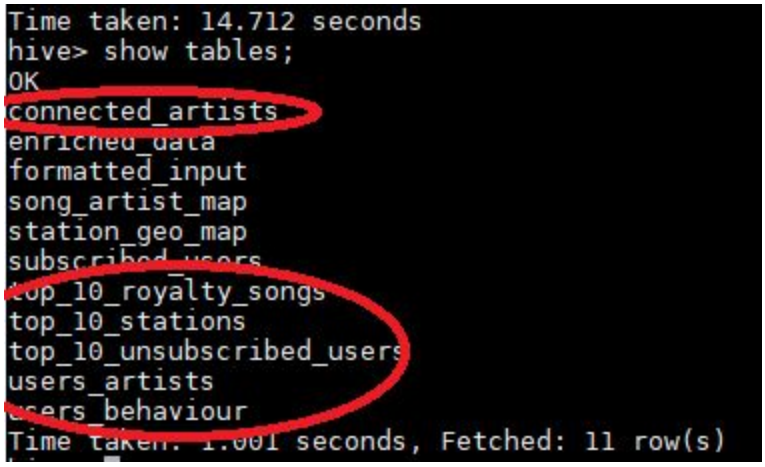
---




**DataAnalysis.sh**

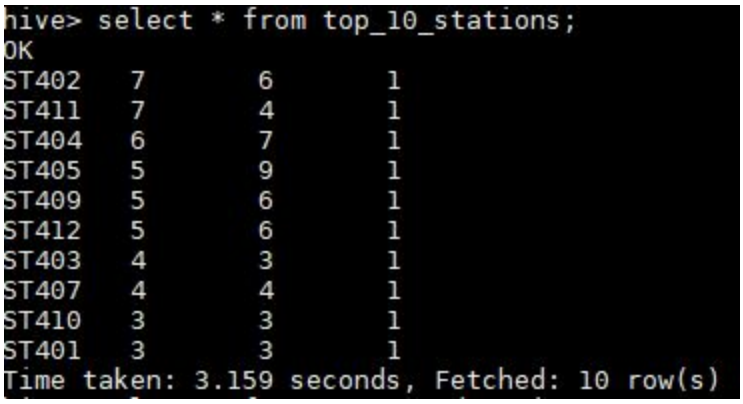
---

DataAnanlysis.scala is used to get the final output using hive query statements on the enriched data table.




## Top 10 stations:

---




## Top 10 Royalty songs:

---

**Top 10 connected artists:**

---

```
hive> select * from connected_artists;
OK
A301    9       1
A302    8       1
A300    3       1
A303    3       1
A304    2       1
A305    1       1
Time taken: 0.358 seconds, Fetched: 6 row(s)
```

**Top 10 unsubscribed users who listened to the songs for the longest duration.**

```
hive> select * from top_10_unsubscribed_users;
OK
U107    228923132       1
U100    167755318       1
U108    138463481       1
U104    109534527       1
U102    103761594       1
U101    98707006        1
U105    78807006        1
U103    62868600        1
U106    62868600        1
U117    52405346        1
Time taken: 0.416 seconds, Fetched: 10 row(s)
```

**User Behaviour:**

---

```
hive> select * from users_behaviour;
OK
UNSUBSCRIBED    1258066757      1
SUBSCRIBED      1833683808      1
Time taken: 0.221 seconds, Fetched: 2 row(s)
```

**Post Data Analysis:**

---

As part of post data analysis, need to export the hive output tables got from previous step into sql tables.

The 2 new scripts created are:

1. Postdatanalysis.sh
2. postanalysis.sql

**Postdataanalysis.sh:**

1.This script creates new tables in sql namely top_10_stations, users_behaviour, connected_artists, top_10_royalty_songs and top_10_unsubscribed_users. This is done using postanalysis.sql script.

2. Using scoop export command gets the data from hdfs//user/hive/metastore/project.db/* and puts into the newly created sql table.

**Snapshot of the tables that are created:**

---

```
mysql> show tables;
+---------------------------+
| Tables_in_project         |
+---------------------------+
| connected_artists         |
| top_10_royalty_songs      |
| top_10_stations           |
| top_10_unsubscribed_users |
| users_behaviour           |
+---------------------------+
5 rows in set (0.00 sec)
```

**Output after the export from hive tables:**

---

**Connected_artists:**

---

```
mysql> select * from connected_artists;
+-----------+------------+
| artist_id | user_count |
+-----------+------------+
| A301      |          9 |
| A302      |          8 |
| A300      |          3 |
| A303      |          3 |
| A304      |          2 |
| A305      |          1 |
+-----------+------------+
6 rows in set (0.00 sec)
```

## Top 10 stations:

---

```
mysql> select * from top_10_stations;
+------------+-----------------------------+---------------------+
| station_id | total_distinct_songs_played | distinct_user_count |
+------------+-----------------------------+---------------------+
| ST404      |                           8 |                   8 |
| ST402      |                           7 |                   6 |
| ST411      |                           7 |                   6 |
| ST409      |                           6 |                   7 |
| ST405      |                           5 |                   9 |
| ST407      |                           5 |                   5 |
| ST412      |                           5 |                   6 |
| ST400      |                           4 |                   4 |
| ST403      |                           4 |                   4 |
| ST410      |                           4 |                   4 |
+------------+-----------------------------+---------------------+
10 rows in set (0.00 sec)
```

## User Behaviour:

---

```
mysql> select * from users_behaviour;
+--------------+------------+
| user_type    | duration   |
+--------------+------------+
| UNSUBSCRIBED | 1433462651 |
| SUBSCRIBED   | 2143723908 |
+--------------+------------+
2 rows in set (0.00 sec)
```

## Top 10 royalty songs:

---

```
mysql> select * from top_10_royalty_songs;
+---------+-----------+
| song_id | duration  |
+---------+-----------+
| S201    | 227952086 |
| S203    | 194298284 |
| S204    | 184402900 |
| S208    | 166266940 |
| S207    | 165273946 |
| S209    | 143008893 |
| S206    | 139311566 |
| S205    | 133131854 |
| S200    | 121697554 |
| S202    |  70286215 |
+---------+-----------+
10 rows in set (0.00 sec)
```

## Top 10 unsubscribed users:

---

```
mysql> select * from top_10_unsubscribed_users;
+---------+-----------+
| user_id | duration  |
+---------+-----------+
| U107    | 257830138 |
| U100    | 221816912 |
| U108    | 160990775 |
| U104    | 109534527 |
| U102    | 103761594 |
| U105    |  98707006 |
| U101    |  98707006 |
| U106    |  82868600 |
| U110    |  66742966 |
| U103    |  62868600 |
+---------+-----------+
10 rows in set (0.00 sec)
```